neers "perform systematic and statistical testing of the software and integrated computer system." Yet, there are no obvious places for usage profiles, risk analysis, design for testability, testware engineering, or team development (estimation, planning, reporting, technical reviews, and so on). This sends a clear message about the priorities of both the author and, even more discouraging, the many acknowledged reviewers.

While I strongly agree with the call to separate software engineering from computer science, I strongly disagree with the proposal to have a software engineering curriculum dominated by design at the exclusion of quality and testing issues. I would rather preserve today's sorry state of affairs than institutionalize this imbalance.

David Gelperin
Software Quality Engineering
sqegelp@aol.com

The software engineering curriculum sketch, "Software Engineering Programs Are Not Computer Science Programs" (Nov./Dec. 1999), by David Lorge Parnas has too little content on software development. It suggests almost all design education with little software development training. It also seems to confuse design and development: "Software Design I: Programming to Meet Precise Specifications" is an oxymoron. I believe there are some necessary items that don't have a place in this software curriculum: life cycles, quality control and management, team organization, and testing. Where do the students learn incremental development and software reliability engineering? Where do they learn inspection and defect prevention?

Tom Adams
Senior Systems Analyst
TRC Environmental Corporation
t.adams@computer.org

David Lorge Parnas replies:

Although my article is quite lengthy, it is still not possible to describe a four-year engineering program in the space available. Thus, both Mr. Gelperin and Mr. Adams were forced to judge McMaster University's curriculum by course title rather than program content.

Mr. Gelperin should be glad to know that we consider testing very important, so important that it is not seen as an "add-on" but as an integral part of every software design course. Testing is also discussed in the statistics course. From the very first project, students are taught the importance of independent testing, test plan preparation, and so on.

Mr. Adams too would probably like the detailed course descriptions more than the titles. Management and teamwork are vital in all engineering areas, not just software. The McMaster program includes a minimum of three substantial team projects, including a senior thesis project that requires a full year. Incremental development is stressed in these projects as are quality control and management. The first course title might have confused Mr. Adams because it is based on our assumption that you must be able to read and use design documents before you learn how to prepare them. Inspection is taught, and there are public inspections during the projects.

If there is disagreement between the letter writers and myself, it is not on the importance of the topics that they mention but on teaching methods. We integrate testing and team building into courses and projects, and they seem to want to treat them in separate courses. In my view that would let the students develop bad habits that we would then have to correct in later courses.

Straightening out the Record

Soheil Khajenoori, in a Nov./Dec. 1999 Letter to the Editor, claims "pioneering status" on the concept of a software engineering body of knowledge. There have indeed been many proposals regarding the software engineering body of knowledge over the past years. Some of them are discussed and referenced in the straw man version of the *Guide to the Software Engineering Body of Knowledge* published in September 1998, which is available at www.swebok.org.

Pierre Bourque
Robert Dupuis
Coeditors, Guide to the Software Engineering Body of Knowledge
Alain Abran
James W. Moore
Coexecutive Editors, Guide to the Software Engineering Body of Knowledge